



# CPSC203 – Introduction to Problem Solving and Using Application Software

Fall 2009

Tutorial 25, Mehrdad Nurolahzade

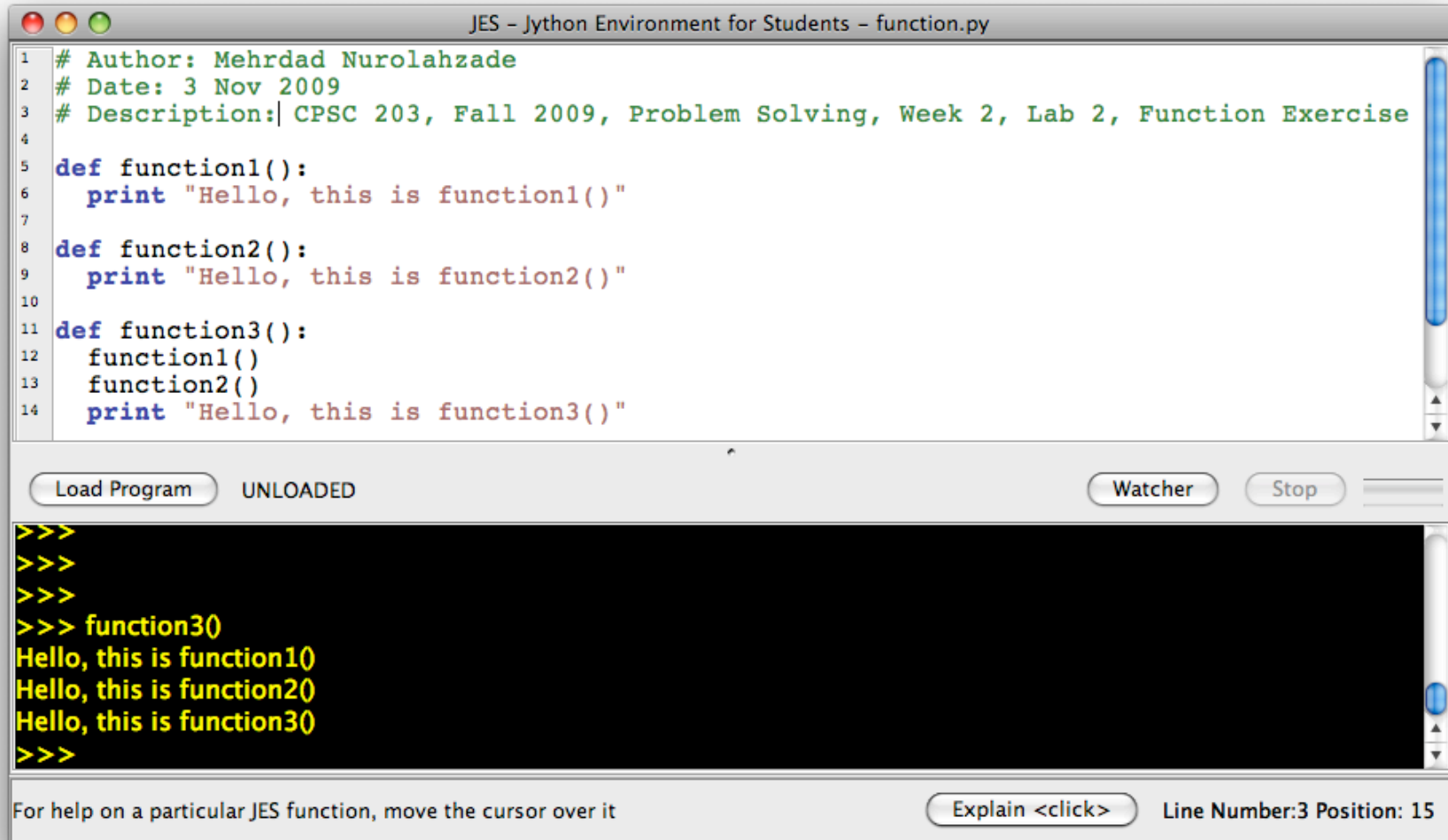
# Introduction

- Functions
- Graphics

# Multiple Functions (1)

- Jython programs can have more than a function.
- A function can call (=invoke) other functions.
- When function A calls function B, all instructions in function B get executed first. Then, once function B returns (=exits) function A continues execution.

# Multiple Functions (2)



The screenshot shows the JES (Jython Environment for Students) interface. The title bar reads "JES - Jython Environment for Students - function.py". The main window contains a Python script with the following code:

```
1 # Author: Mehrdad Nurolahzade
2 # Date: 3 Nov 2009
3 # Description: CPSC 203, Fall 2009, Problem Solving, Week 2, Lab 2, Function Exercise
4
5 def function1():
6     print "Hello, this is function1()"
7
8 def function2():
9     print "Hello, this is function2()"
10
11 def function3():
12     function1()
13     function2()
14     print "Hello, this is function3()"
```

Below the code editor, there are buttons for "Load Program", "UNLOADED", "Watcher", and "Stop". The "Load Program" button is highlighted. The output window shows the following execution results:

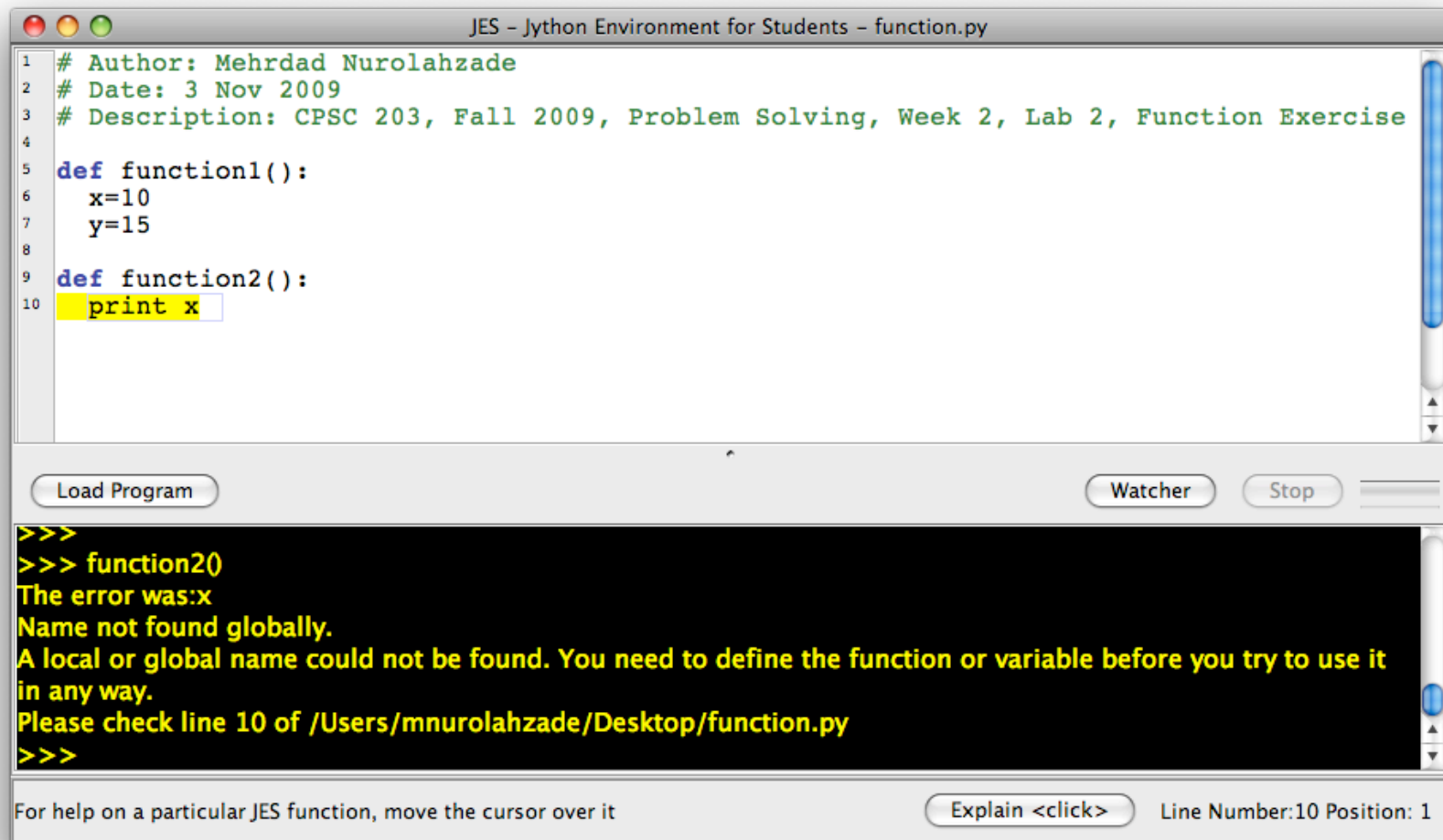
```
>>>
>>>
>>>
>>> function3()
Hello, this is function1()
Hello, this is function2()
Hello, this is function3()
>>>
```

At the bottom of the interface, there is a status bar with the text "For help on a particular JES function, move the cursor over it" and a button labeled "Explain <click>". The status bar also displays "Line Number:3 Position: 15".

# Variable Scope (1)

- All variables defined inside the body of a function are local to that function, i.e. cannot be accessed in other functions.

# Variable Scope (2)



The screenshot shows the JES IDE window titled "JES - Jython Environment for Students - function.py". The editor contains the following Python code:

```
1 # Author: Mehrdad Nurolahzade
2 # Date: 3 Nov 2009
3 # Description: CPSC 203, Fall 2009, Problem Solving, Week 2, Lab 2, Function Exercise
4
5 def function1():
6     x=10
7     y=15
8
9 def function2():
10  print x
```

Below the editor, there are buttons for "Load Program", "Watcher", and "Stop". The console output shows the execution of `function2()` and an error message:

```
>>>
>>> function2()
The error was:x
Name not found globally.
A local or global name could not be found. You need to define the function or variable before you try to use it
in any way.
Please check line 10 of /Users/mnurolahzade/Desktop/function.py
>>>
```

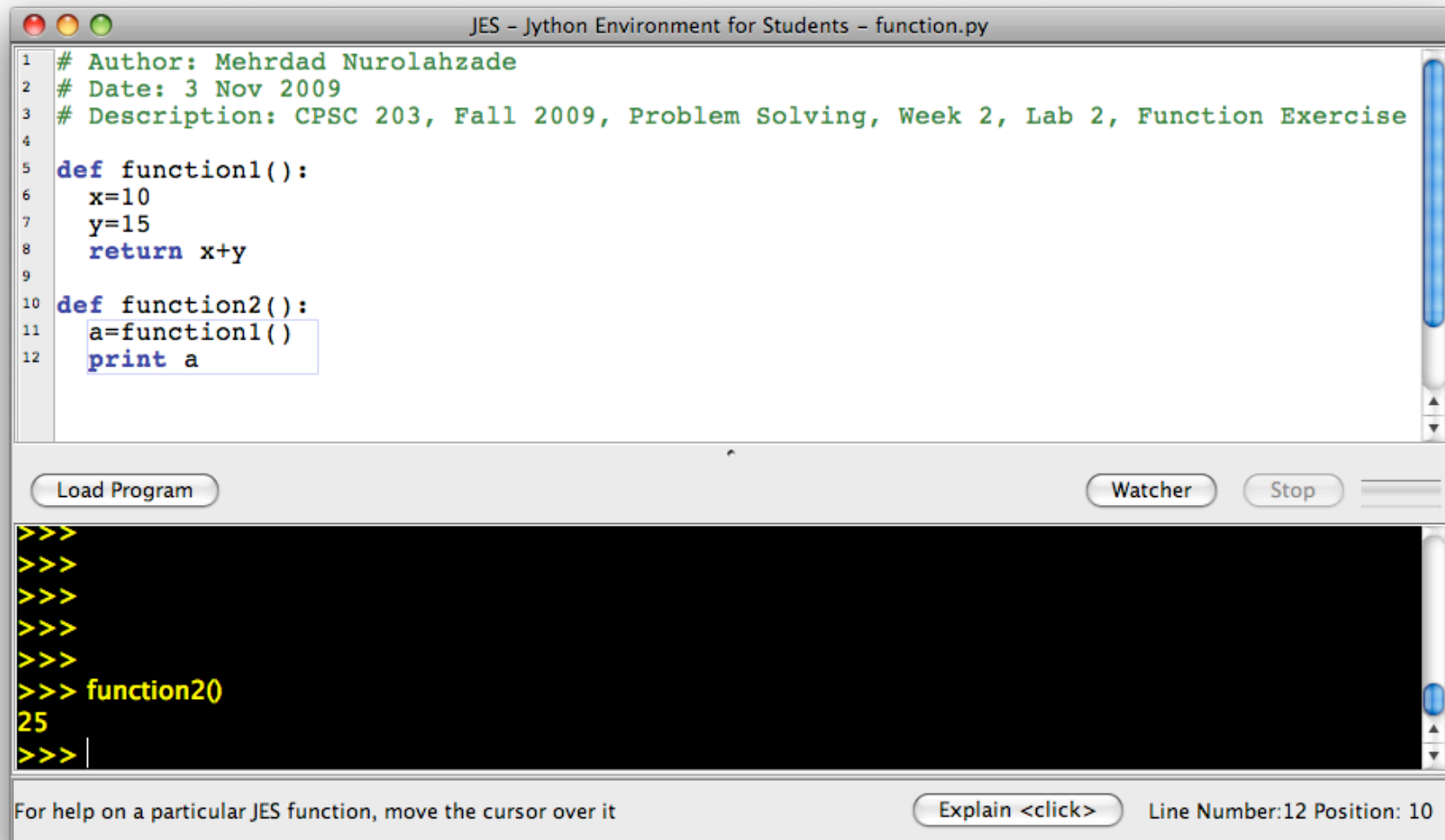
At the bottom of the window, there is a status bar with the text "For help on a particular JES function, move the cursor over it" and a button labeled "Explain <click>". To the right of the button, it says "Line Number:10 Position: 1".

# Return Value (1)

- A function can return a value upon exit.
- Function A calling function B receives its return value.
- **General syntax:**

```
def function-name(zero or more arguments):  
    statement1  
    statement2  
    ...  
    statementN  
    return return-value
```

# Return Value (2)



The screenshot shows the JES IDE window titled "JES - Jython Environment for Students - function.py". The editor contains the following Python code:

```
1 # Author: Mehrdad Nurolahzade
2 # Date: 3 Nov 2009
3 # Description: CPSC 203, Fall 2009, Problem Solving, Week 2, Lab 2, Function Exercise
4
5 def function1():
6     x=10
7     y=15
8     return x+y
9
10 def function2():
11     a=function1()
12     print a
```

Below the editor, there are buttons for "Load Program", "Watcher", and "Stop". The console window shows the execution of the code:

```
>>>
>>>
>>>
>>>
>>> function2()
25
>>> |
```

At the bottom of the window, there is a status bar with the text "For help on a particular JES function, move the cursor over it" and a button labeled "Explain <click>". The status bar also displays "Line Number:12 Position: 10".



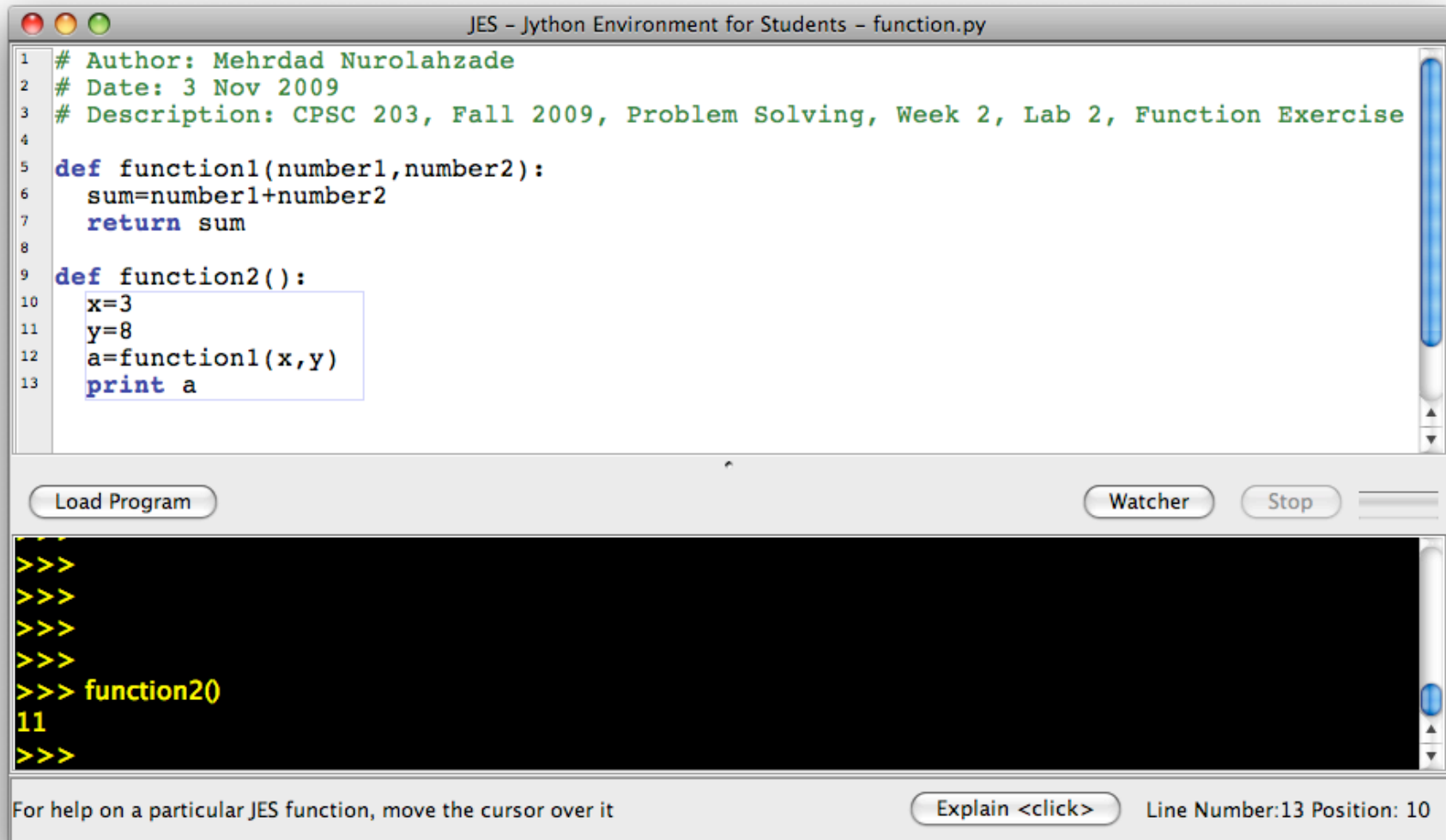
# Function Arguments (1)

- Function arguments are a list of parameters that are passed to the function.
- A function can have zero or more arguments.

- **General syntax:**

```
def function-name(arg1, arg2, ..., argN) :  
    statements  
    return return-value
```

# Function Arguments (2)



The screenshot shows the JES IDE window titled "JES - Jython Environment for Students - function.py". The editor contains the following Python code:

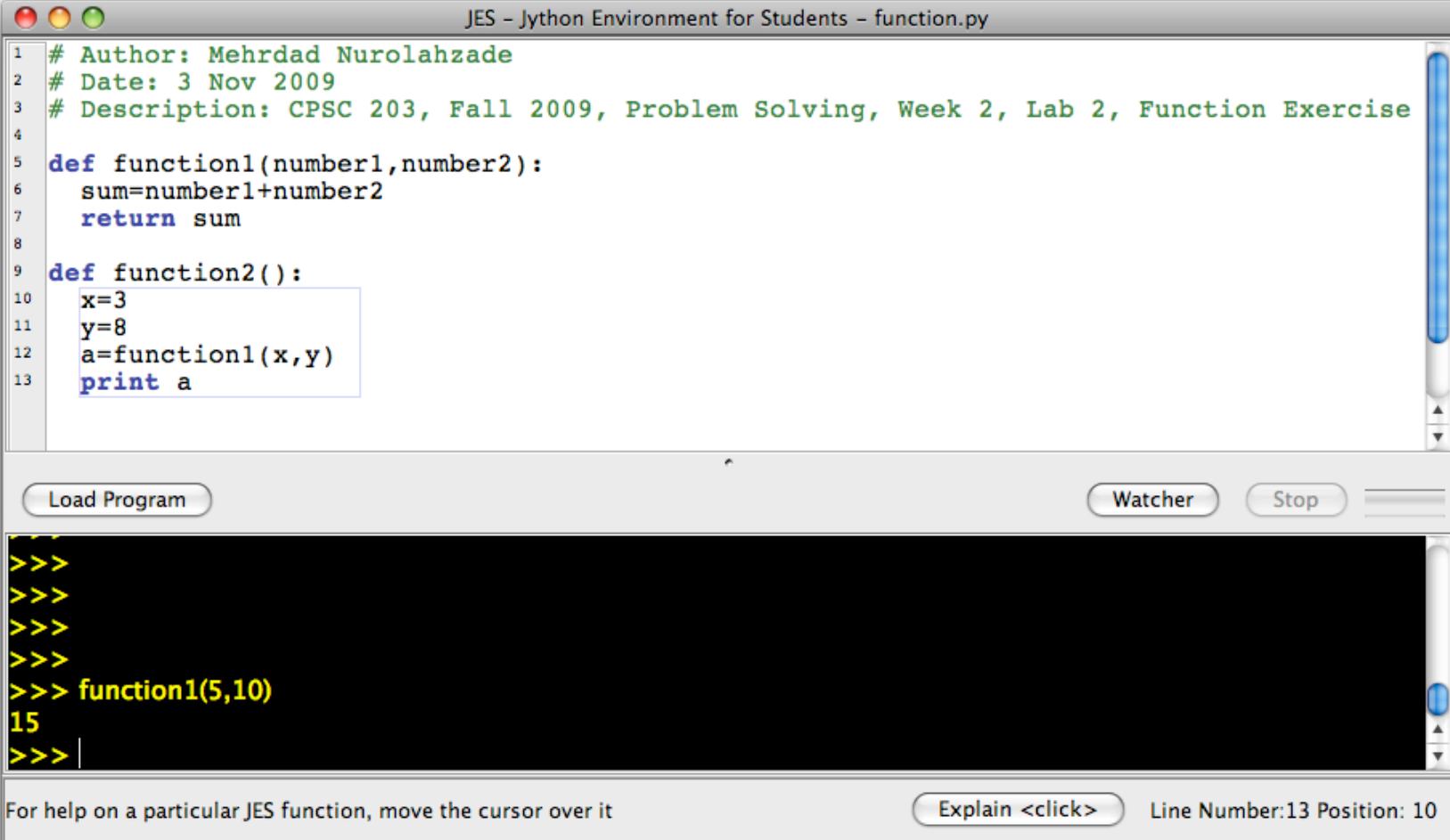
```
1 # Author: Mehrdad Nurolahzade
2 # Date: 3 Nov 2009
3 # Description: CPSC 203, Fall 2009, Problem Solving, Week 2, Lab 2, Function Exercise
4
5 def function1(number1,number2):
6     sum=number1+number2
7     return sum
8
9 def function2():
10     x=3
11     y=8
12     a=function1(x,y)
13     print a
```

Below the editor is a toolbar with buttons for "Load Program", "Watcher", and "Stop". The console window at the bottom shows the execution of the code:

```
>>>
>>>
>>>
>>> function2()
11
>>>
```

At the bottom of the IDE, there is a status bar with the text "For help on a particular JES function, move the cursor over it" and a button labeled "Explain <click>". To the right of this button, it says "Line Number:13 Position: 10".

# Function Arguments (3)



The screenshot shows the JES (Jython Environment for Students) interface. The top window displays a Python script named `function.py`. The script defines two functions: `function1` and `function2`. `function1` takes two arguments, `number1` and `number2`, and returns their sum. `function2` calls `function1` with arguments `x=3` and `y=8` and prints the result. The bottom window shows the execution of the script, with the prompt `>>>` and the function call `function1(5,10)` resulting in the output `15`. The status bar at the bottom indicates the current line number is 13 and the position is 10.

```
1 # Author: Mehrdad Nurolahzade
2 # Date: 3 Nov 2009
3 # Description: CPSC 203, Fall 2009, Problem Solving, Week 2, Lab 2, Function Exercise
4
5 def function1(number1,number2):
6     sum=number1+number2
7     return sum
8
9 def function2():
10     x=3
11     y=8
12     a=function1(x,y)
13     print a
```

Load Program Watcher Stop

```
>>>
>>>
>>>
>>> function1(5,10)
15
>>> |
```

For help on a particular JES function, move the cursor over it Explain <click> Line Number:13 Position: 10

# Function Exercise (1)

- Write the Jython function **count(n)** that given the argument **n** of type Integer prints out numbers from 1 to **n**.

- Example run:

```
count(4)
```

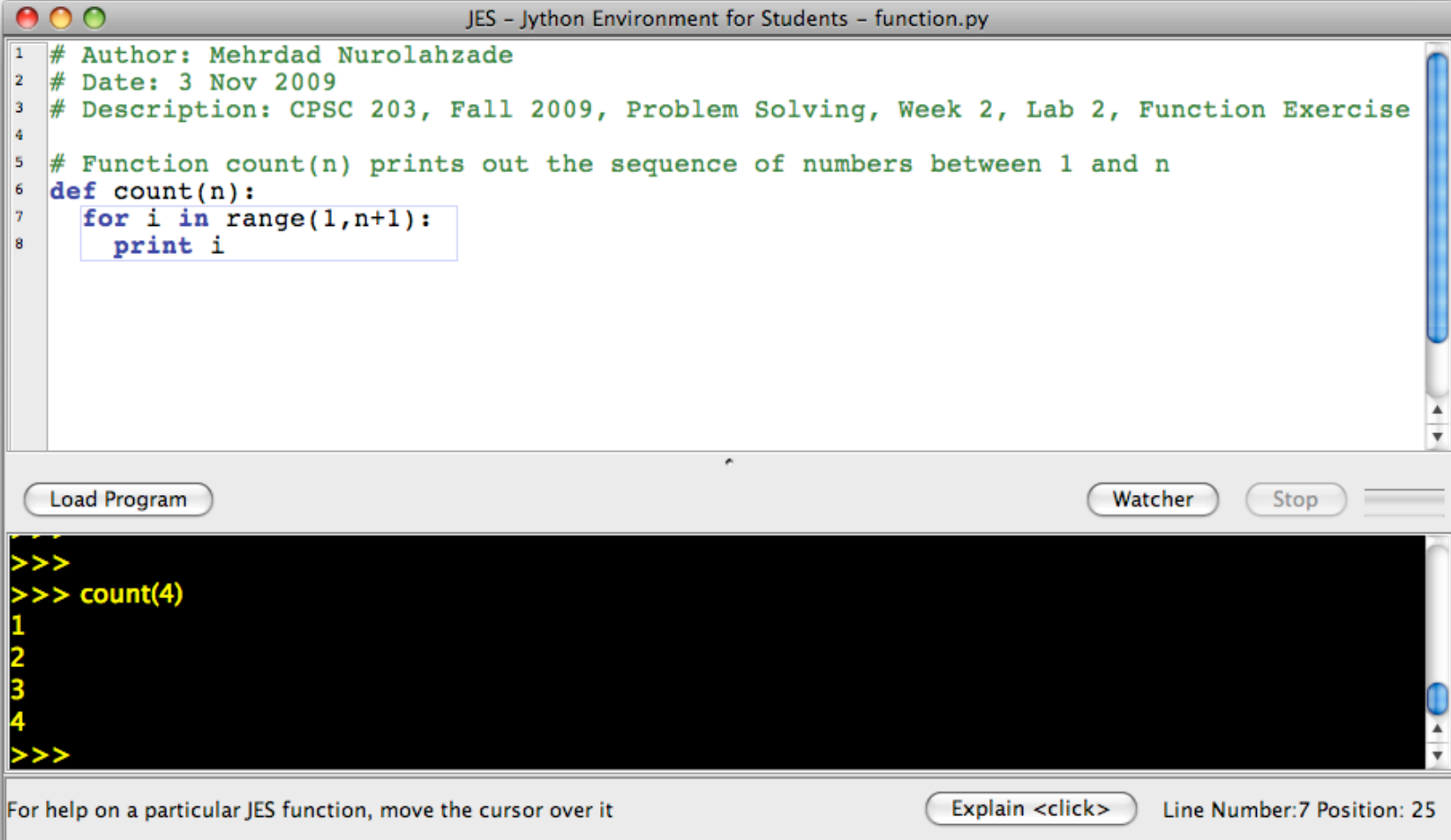
```
1
```

```
2
```

```
3
```

```
4
```

# Function Exercise (2)



The screenshot shows the JES (Jython Environment for Students) interface. The top window displays a Python script named `function.py` with the following content:

```
1 # Author: Mehrdad Nurolahzade
2 # Date: 3 Nov 2009
3 # Description: CPSC 203, Fall 2009, Problem Solving, Week 2, Lab 2, Function Exercise
4
5 # Function count(n) prints out the sequence of numbers between 1 and n
6 def count(n):
7     for i in range(1,n+1):
8         print i
```

Below the script editor, there are buttons for `Load Program`, `Watcher`, and `Stop`. The bottom panel shows the execution of the `count(4)` function, resulting in the output:

```
>>>
>>> count(4)
1
2
3
4
>>>
```

At the bottom of the interface, there is a status bar with the text "For help on a particular JES function, move the cursor over it" and a button labeled `Explain <click>`. The status bar also displays "Line Number:7 Position: 25".

## Function Exercise (3)

- Write the Python function **minimum(S)** that finds the minimum element in the given list **S** of Integers.
- Note: you are not allowed to use the **min()** function.

- Example run:

```
minimum([7, 3, 5, 6, 4, 2, 3])  
2
```

## Function Exercise (4)

```
def minimum(S):  
    if len(S)==0:  
        return 'undefined'  
    else:  
        min_so_far = S[0]  
        for i in range(1,len(S)):  
            if S[i] < min_so_far:  
                min_so_far = S[i]  
        return min_so_far
```

## Function Exercise (5)

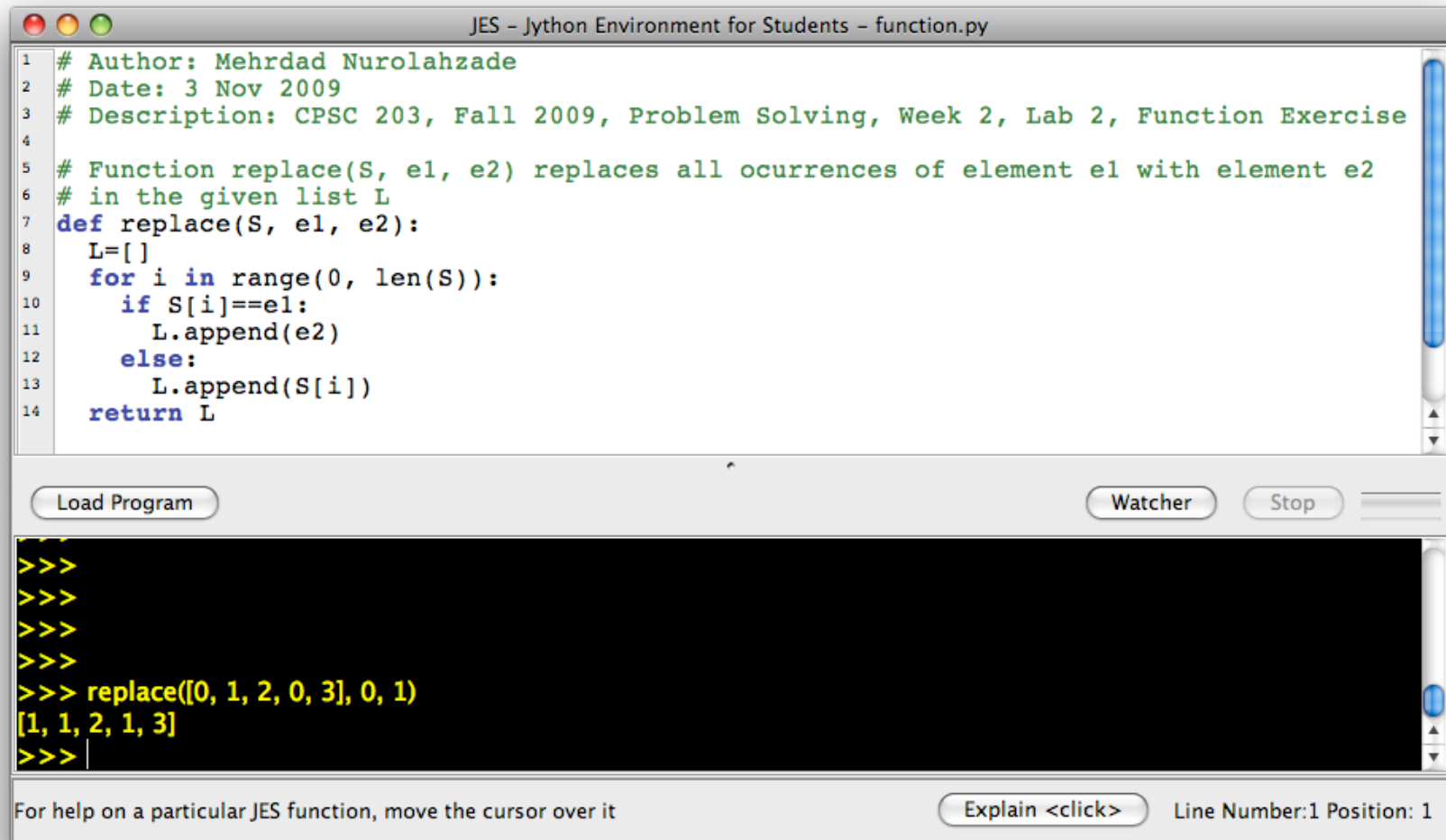
- Write the Jython function **replace(L, e1, e2)** that replaces all occurrences of element **e1** with element **e2** in the given list **L**.

- Example run:

```
replace(['a', 'd', 'a', 'c', 'b'], 'a', 'b')  
['b', 'd', 'b', 'c', 'b']
```



# Function Exercise (6)



The screenshot shows the JES (Jython Environment for Students) interface. The top window displays a Python script named 'function.py'. The script defines a function 'replace(S, e1, e2)' that iterates through a list 'S' and replaces all occurrences of 'e1' with 'e2', returning a new list 'L'. Below the script editor, there are buttons for 'Load Program', 'Watcher', and 'Stop'. The bottom window shows the interactive prompt with the function call 'replace([0, 1, 2, 0, 3], 0, 1)' and its output '[1, 1, 2, 1, 3]'. At the bottom of the interface, there is a status bar with the text 'For help on a particular JES function, move the cursor over it', a button labeled 'Explain <click>', and the text 'Line Number:1 Position: 1'.

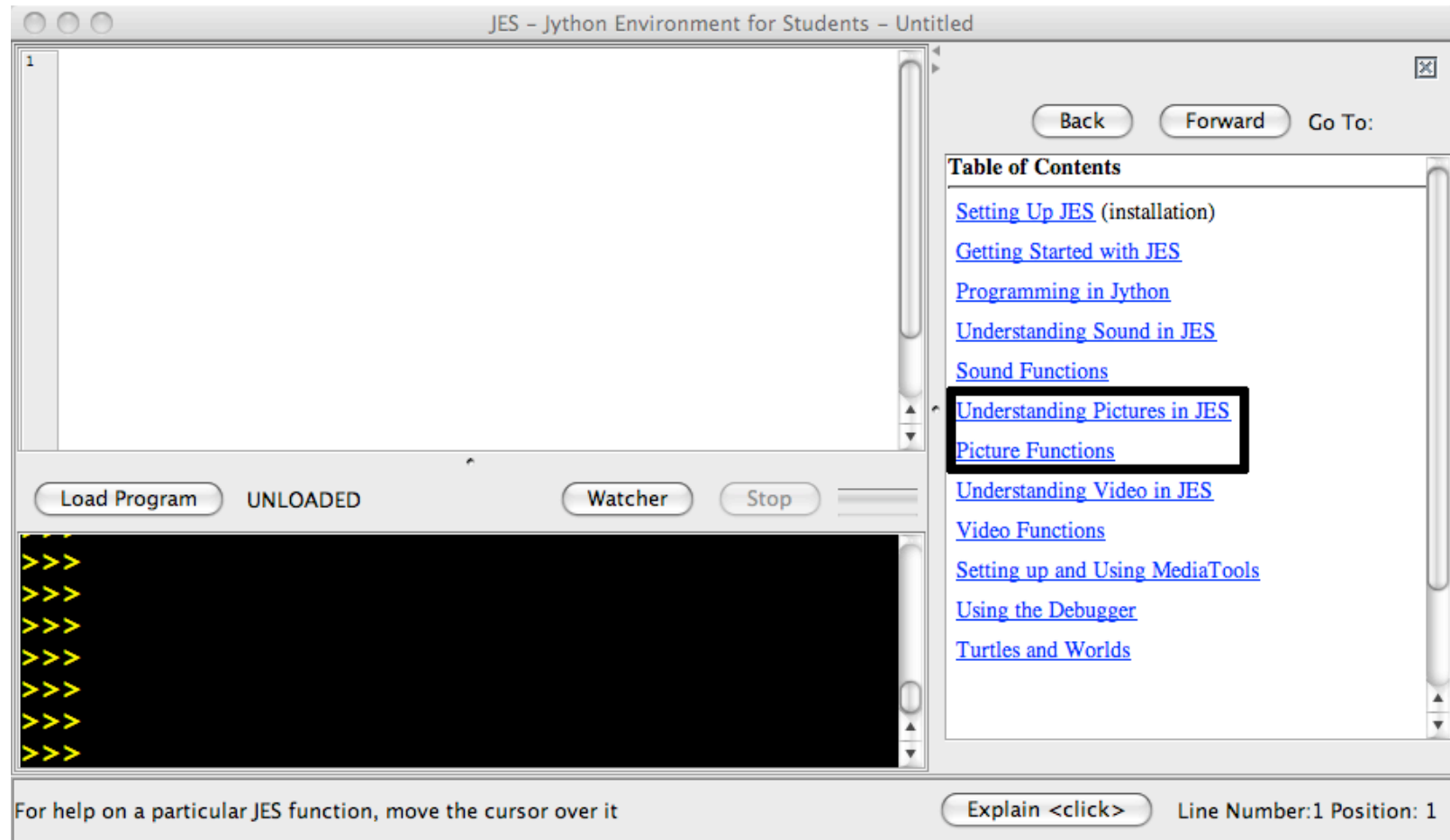
```
1 # Author: Mehrdad Nurolahzade
2 # Date: 3 Nov 2009
3 # Description: CPSC 203, Fall 2009, Problem Solving, Week 2, Lab 2, Function Exercise
4
5 # Function replace(S, e1, e2) replaces all occurrences of element e1 with element e2
6 # in the given list L
7 def replace(S, e1, e2):
8     L=[]
9     for i in range(0, len(S)):
10         if S[i]==e1:
11             L.append(e2)
12         else:
13             L.append(S[i])
14     return L
```

Load Program Watcher Stop

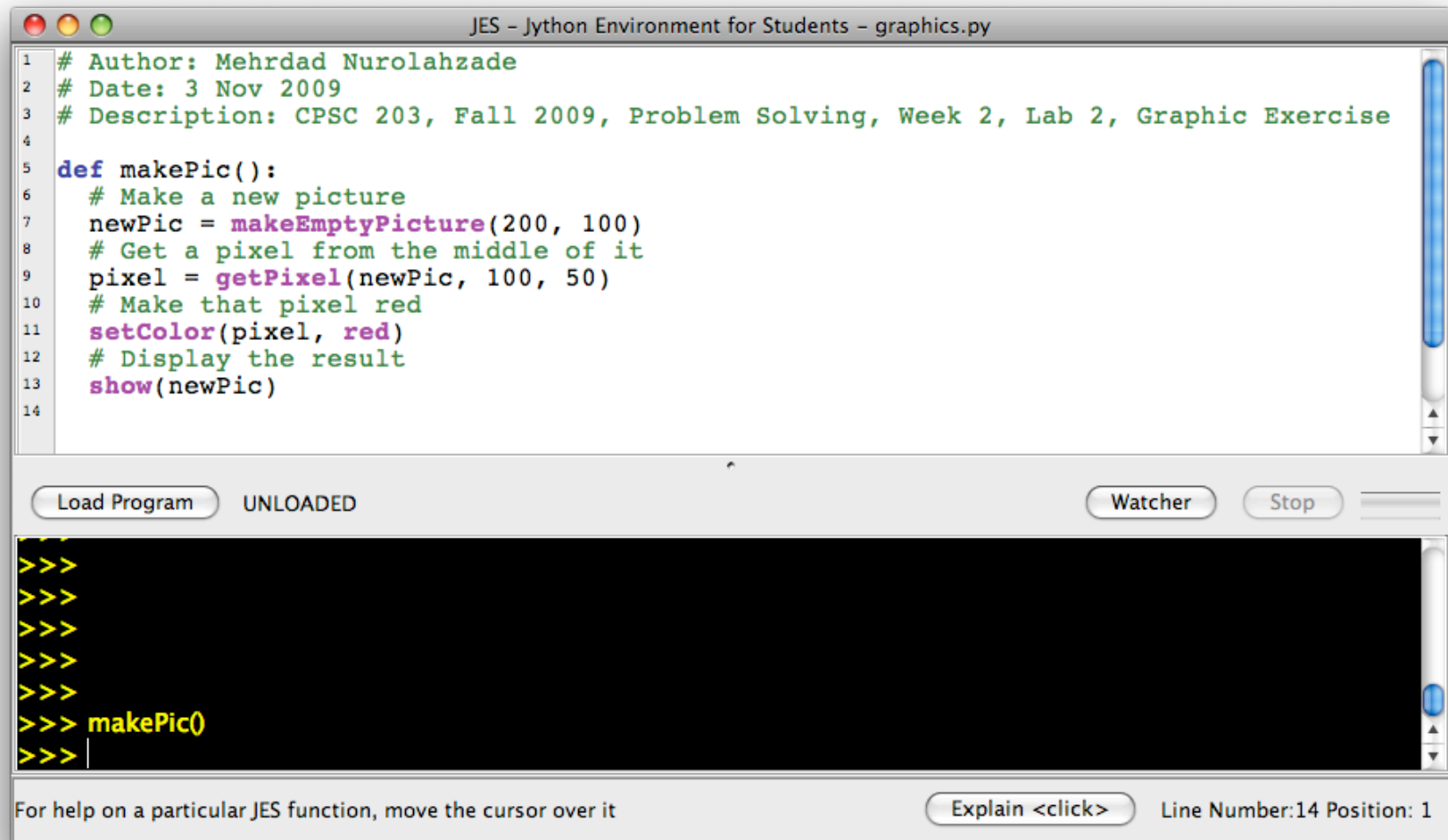
```
>>>
>>>
>>>
>>> replace([0, 1, 2, 0, 3], 0, 1)
[1, 1, 2, 1, 3]
>>>
```

For help on a particular JES function, move the cursor over it Explain <click> Line Number:1 Position: 1

# Graphics in Jython



# Making A Red Dot



The screenshot shows the JES (Jython Environment for Students) interface. The top window displays a Python script for creating a red dot. The script includes comments about the author (Mehrdad Nurolahzade), date (3 Nov 2009), and description (CPSC 203, Fall 2009, Problem Solving, Week 2, Lab 2, Graphic Exercise). The code defines a function `makePic()` that creates a new picture, gets a pixel from the middle, sets its color to red, and displays the result. The bottom window shows the command prompt with the `makePic()` function being called. The status bar at the bottom indicates the current line number is 14 and the position is 1.

```
1 # Author: Mehrdad Nurolahzade
2 # Date: 3 Nov 2009
3 # Description: CPSC 203, Fall 2009, Problem Solving, Week 2, Lab 2, Graphic Exercise
4
5 def makePic():
6     # Make a new picture
7     newPic = makeEmptyPicture(200, 100)
8     # Get a pixel from the middle of it
9     pixel = getPixel(newPic, 100, 50)
10    # Make that pixel red
11    setColor(pixel, red)
12    # Display the result
13    show(newPic)
14
```

Load Program UNLOADED Watcher Stop

>>>  
>>>  
>>>  
>>>  
>>>  
>>> makePic()  
>>>

For help on a particular JES function, move the cursor over it Explain <click> Line Number:14 Position: 1

# makeEmptyPicture()

**makeEmptyPicture**(width, height[, color]):

width: the width of the empty picture

height: height of the empty picture

color: background color of the empty picture (optional)

returns: a new picture object with all the pixels set to the specified color

Makes a new "empty" picture and returns it to you. The width and height must be between 0 and 10000. Default color is white.

# getPixel()

**getPixel**(picture, xpos, ypos):

picture: the picture you want to get the pixel from

xpos: the x-coordinate of the pixel you want

ypos: the y-coordinate of the pixel you want

Takes a picture, an x position and a y position (two numbers),  
and returns the Pixel object at that point in the picture.

# setColor()

**setColor(pixel, color):**

pixel: the pixel you want to set the color of

color: the color you want to set the pixel to

Takes in a pixel and a color, and sets the pixel to the given color.

# show()

**show**(picture):

picture: the picture you want to see

Shows the picture provided as input.

# addLine()

**addLine**(picture, startX, startY, endX, endY[, color]):

picture: the picture you want to draw the line on

startX: the x position you want the line to start

startY: the y position you want the line to start

endX: the x position you want the line to end

endY: the y position you want the line to end

color: the color you want to draw in (optional)

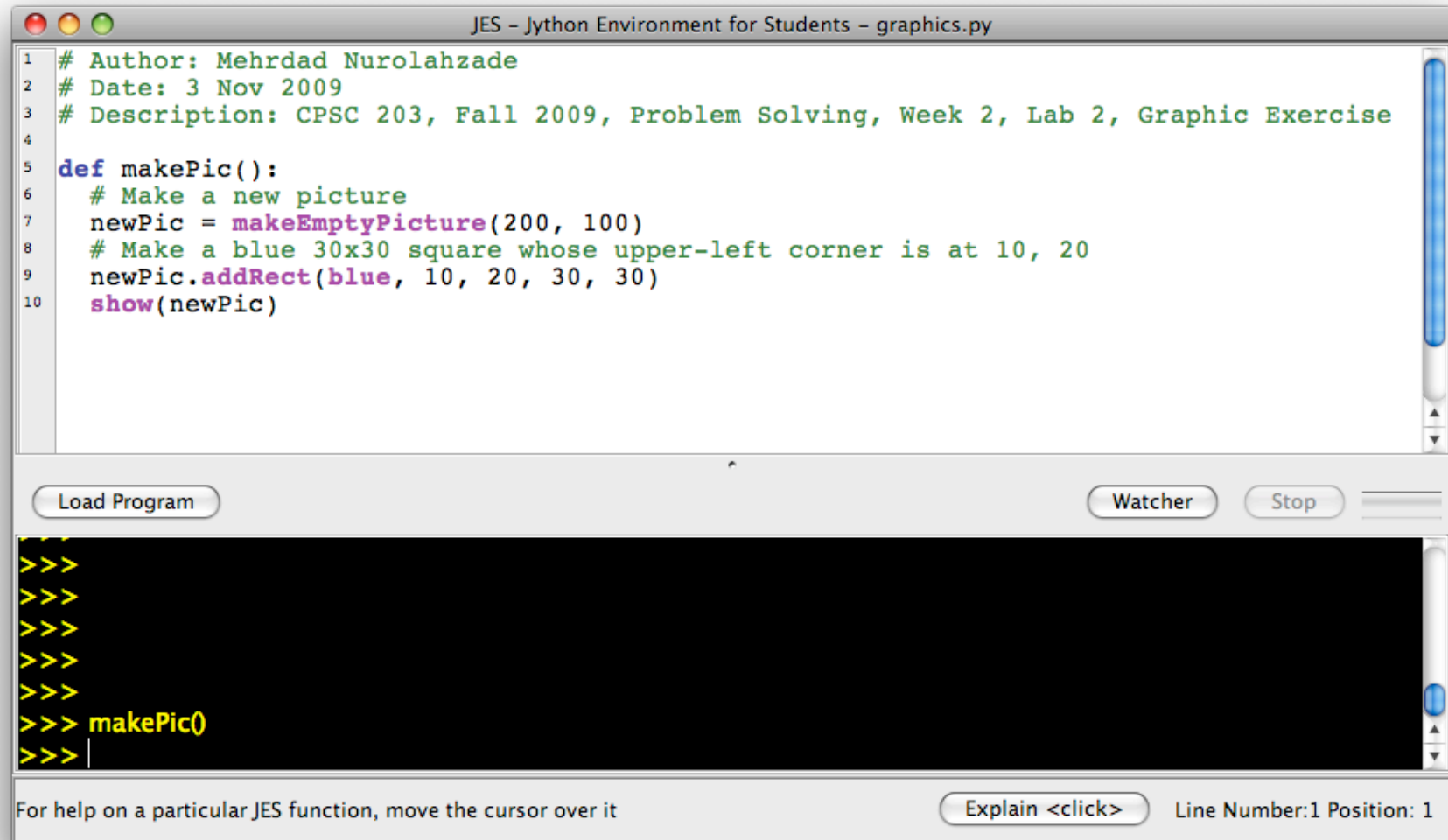
Takes a picture, a starting (x, y) position (two numbers), and an ending (x, y) position (two more numbers, four total), and (optionally) a color as input.

Adds a line from the starting point to the ending point in the picture.

Default color is black.



# Draw A Blue Rectangle



```
1 # Author: Mehrdad Nurolahzade
2 # Date: 3 Nov 2009
3 # Description: CPSC 203, Fall 2009, Problem Solving, Week 2, Lab 2, Graphic Exercise
4
5 def makePic():
6     # Make a new picture
7     newPic = makeEmptyPicture(200, 100)
8     # Make a blue 30x30 square whose upper-left corner is at 10, 20
9     newPic.addRect(blue, 10, 20, 30, 30)
10    show(newPic)
```

>>>  
>>>  
>>>  
>>>  
>>>  
>>> makePic()  
>>>

For help on a particular JES function, move the cursor over it Explain <click> Line Number:1 Position: 1

# addRect()

**addRect**(picture, startX, startY, width, height[, color]):

picture: the picture you want to draw the rectangle on

startX: the x-coordinate of the upper left-hand corner of the rectangle

startY: the y-coordinate of the upper left-hand corner of the rectangle

width: the width of the rectangle

height: the height of the rectangle

color: the color you want to draw in (optional)

Takes a picture, a starting (x, y) position (two numbers), a width and height (two more numbers, four total), and (optionally) a color as input. Adds a rectangular outline of the specified dimensions using the (x,y) as the upper left corner. Default color is black.

# addOval()

**addOval**(picture, startX, startY, width, height[, color]):

picture: the picture you want to draw the rectangle on

startX: the x-coordinate of the upper left-hand corner of the bounding rectangle of the oval

startY: the y-coordinate of the upper left-hand corner of the bounding rectangle of the oval

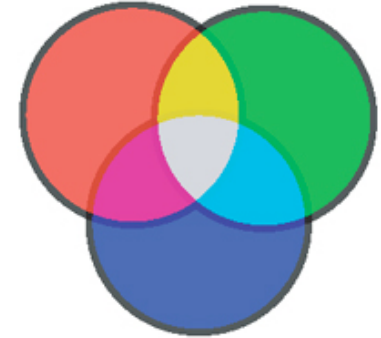
width: the width of the oval

height: the height of the oval

color: the color you want to draw in (optional)

Takes a picture, a starting (x, y) position (two numbers), a width and height (two more numbers, four total), and (optionally) a color as input. Adds an oval outline of the given dimensions using the (x,y) as the upper left corner of the bounding rectangle. Default color is black.

# makeColor()



**makeColor**(red[, green, blue]):

red: the amount of red you want in the color (or a Color object you want to duplicate)

green: the amount of green you want in the color (optional)

blue: the amount of blue you want in the picture (optional)

returns: the color made from the inputs

Takes three integer inputs for the red, green, and blue components (in order) and returns a color object. If green and blue are omitted, the red value is used as the intensity of a gray color. Also it works with only a color as input and returns a new color object with the same RGB values as the original.

# addText()

**addText**(picture, xpos, ypos, text[, color]):

picture: the picture you want to add the text to

xpos: the x-coordinate where you want to start writing the text

ypos: the y-coordinate where you want to start writing the text

text: string containing the text you want written

color: the color you want to draw in (optional)

Takes a picture, an x position and a y position (two numbers), and some text as a string, which will get drawn into the picture, in the specified color. Default is black.

# writePictureTo()

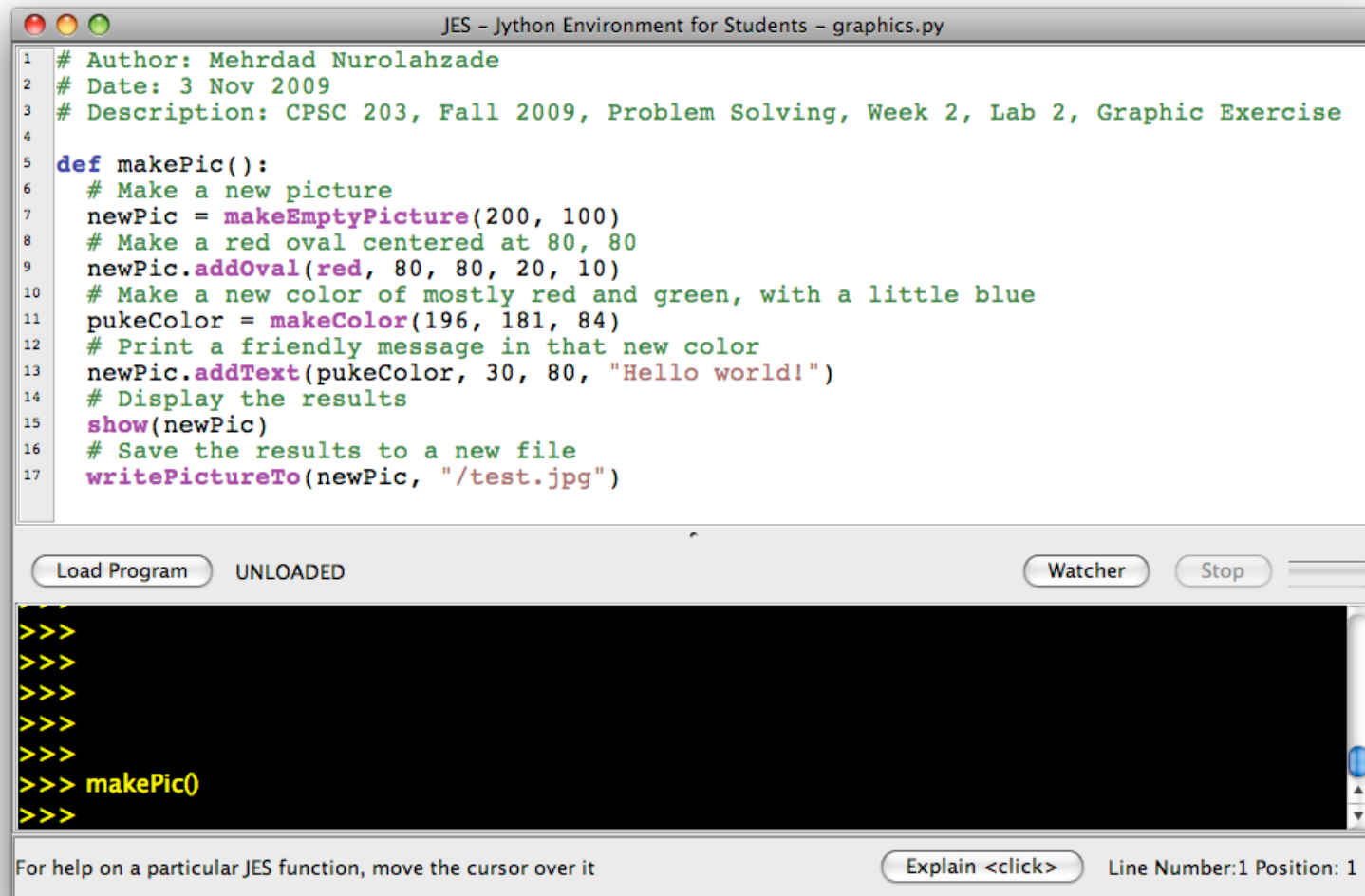
**writePictureTo**(picture, path):

picture: the picture you want to be written out to a file

path: the path to the file you want the picture written to

Takes a picture and a file name (string) as input, then writes the picture to the file as a JPEG, PNG, or BMP. (Be sure to end the filename in ".jpg" or ".png" or ".bmp" for the operating system to understand it well.)

# Graphics Example



The screenshot shows the JES (Jython Environment for Students) interface. The title bar reads "JES - Jython Environment for Students - graphics.py". The main window contains a Python script with the following code:

```
1 # Author: Mehrdad Nurolahzade
2 # Date: 3 Nov 2009
3 # Description: CPSC 203, Fall 2009, Problem Solving, Week 2, Lab 2, Graphic Exercise
4
5 def makePic():
6     # Make a new picture
7     newPic = makeEmptyPicture(200, 100)
8     # Make a red oval centered at 80, 80
9     newPic.addOval(red, 80, 80, 20, 10)
10    # Make a new color of mostly red and green, with a little blue
11    pukeColor = makeColor(196, 181, 84)
12    # Print a friendly message in that new color
13    newPic.addText(pukeColor, 30, 80, "Hello world!")
14    # Display the results
15    show(newPic)
16    # Save the results to a new file
17    writePictureTo(newPic, "/test.jpg")
```

Below the code editor, there is a "Load Program" button, a status indicator "UNLOADED", a "Watcher" button, and a "Stop" button. The bottom section is a black console area with yellow text showing the execution of the `makePic()` function. At the very bottom, there is a status bar with the text "For help on a particular JES function, move the cursor over it", a button labeled "Explain <click>", and a status indicator "Line Number:1 Position: 1".